Лабораторная работа №2

Моделирование схем в Verilog

Цель работы: построение кэша и моделирование системы “процессор-кэш-память” на языке описания Verilog.

Инструментарий: весь код пишется на языке Verilog, компиляция и симуляция – Icarus Verilog 10 и новее (полезные материалы: Verilog.docx). В отчёте нужно указать, какой версией вы пользовались (можно также приложить ссылку на онлайн-платформу). Использовать SystemVerilog допустимо, главное, чтобы код компилился под Icarus 10, 11 или 12. Далее в этом документе Verilog+SystemVerilog обозначается как Verilog.

Порядок сдачи работы:

1. Выполнить работу.
2. Оформить отчет в формате pdf.
3. Загрузить файл отчета и файлы с модулями и тестовым окружением (расширение *.sv) на github в выданный вам репозиторий.
4. Запустить автотесты. Подробнее: ЭВМ осень 2022. Github (ЛР2).

Шаблон отчета и порядок отправки работ на проверку

ЭВМ осень 2022. Порядок сдачи лабораторных работ

Содержание отчета

1. Минититульник (таблица с ФИО и названием работы из шаблона).
2. Цель работы.
3. Инструментарий.
4. Формулировка задачи из условия.
5. Вычисление недостающих параметров системы.
6. Аналитическое решение задачи (решение вида написать код на языке высокого уровня, эмулирующего работу системы, будет засчитано).

7. Моделирование заданной системы на Verilog.
8. Воспроизведение задачи на Verilog.
9. Сравнение полученных результатов.
10. Листинг кода

Пояснения:

В качестве варианта вам даны некоторые параметры системы и задача, которую надо сначала решить аналитически на основании параметров системы, а затем промоделировать и сравнить полученные результаты.

В *пункте 7* опишите, как работает выданная вам модель. Какие в ней составные элементы (которые в последующем становятся модулями), что они из себя представляют (например, для кэша заданы политики чтения, записи, замещения - нужно расписать как они работают). Как вы описали систему из условия средствами языка описания аппаратуры Verilog. Как они работают (когда срабатывает та или иная операция, когда происходит чтение, запись данных). Можно иллюстрировать пояснения частичными листингами кода (например, нужный блок always показать и пояснить, что он делает).

Пункт 8. В тестовом окружении вы должны промоделировать задачу. Здесь желательно приводить временную диаграмму или лог с выводом времени.

В *пункте 10* приведите листинг кода (даже несмотря на то, что код вы и так сдаете отдельными файлами, в этот раздел нужно его вставить текстом).

Система “процессор-кэш-память”

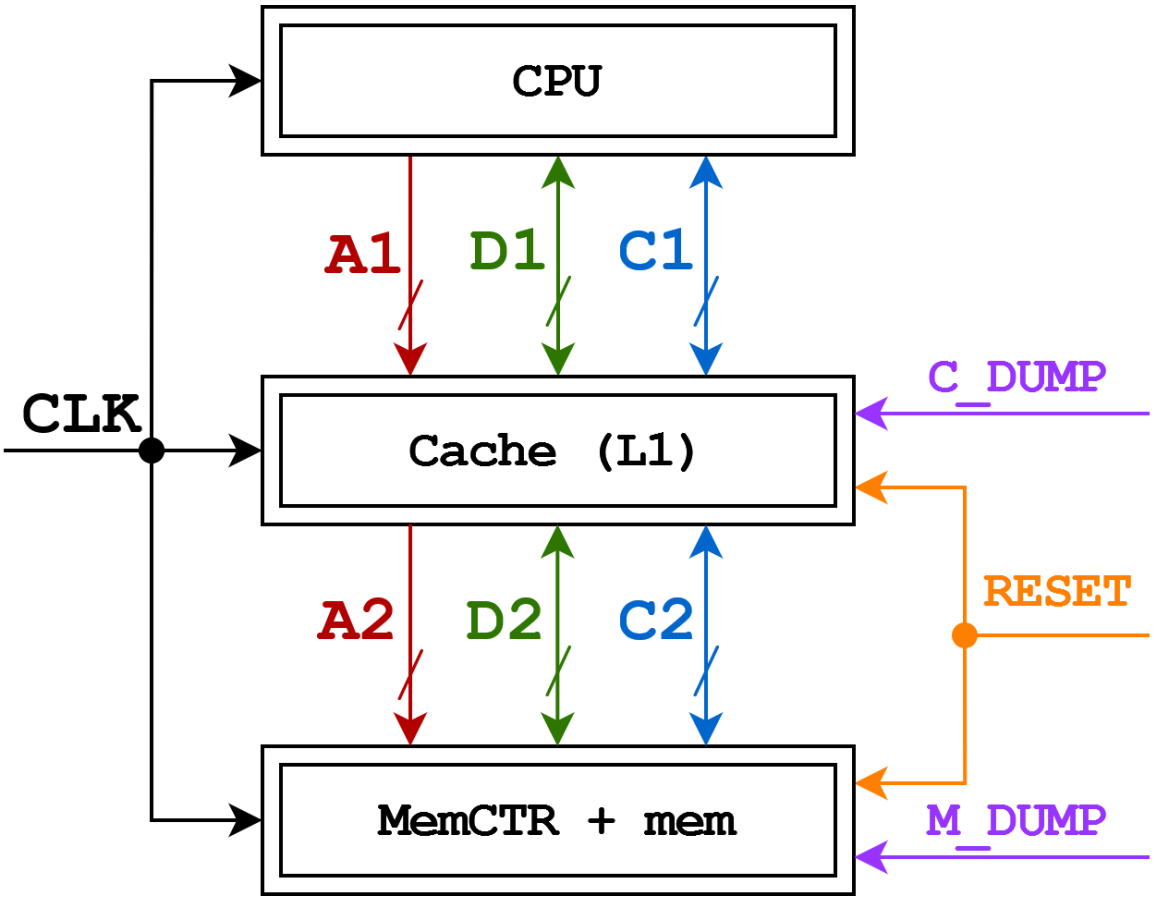

Рисунок 1 – Схема моделируемой системы

Условные обозначения:

`CACHE_SIZE` – названия переменных/параметров, которые рекомендуется использовать в работе. Полный список в разделе Обозначения параметров в коде.

Сигналы:

- **`CLK`** – синхронизация всей схемы
- **`RESET`** – сброс в начальное состояние
- **`*_DUMP`** – сохранение текущего состояния в файл/вывод в консоль для отладки

Модули:

- CPU – модель процессора для верификации работы кэша
- Cache – одноуровневый кэш

- MemCTR (+ mem) – модель контроллера памяти + модулей памяти для верификации работы кэша

valid	dirty	tag	data
1	1	CACHE_TAG_SIZE	CACHE_LINE_SIZE

Рисунок 2 – Устройство кэш-линии

tag	set	offset
CACHE_TAG_SIZE	CACHE_SET_SIZE	CACHE_OFFSET_SIZE

Рисунок 3 – Интерпретация адреса кэшем

Везде, где не указаны значения, означает, что их нужно вычислить в ходе выполнения работы. Всего есть 3 варианта, различающихся параметрами кэша.

Параметры системы (общие)

CPU			
Команды	CPU → Cache	0 – **C1_NOP** 1 – **C1_READ8** 2 – **C1_READ16** 3 – **C1_READ32** 4 – **C1_INVALIDATE_LINE** 5 – **C1_WRITE8** 6 – **C1_WRITE16** 7 – **C1_WRITE32**	Команда 4 означает инвалидацию всей кэш-линии, содержащей указанный адрес. Число в командах означает кол-во бит данных, запрашиваемое данной командой. Команды, запрашивающие несколько байт, не могут пересекать кэш-линию.
	CPU ← Cache	0 – **C1_NOP** 7 – **C1_RESPONSE**	

			NOP – no operation. Response – ответ на команду.
Кэш (look-through write-back)			
Политика вытеснения		LRU	
Команды	Cache → Mem	0 – **C2_NOP** 2 – **C2_READ_LINE** 3 – **C2_WRITE_LINE**	Команды пишут и читают порциями, равными размеру кэш-линии.
	Cache ← Mem	0 – **C2_NOP** 1 – **C2_RESPONSE**	
Служебные биты		V (valid), D (dirty)	Если valid установлен в 0, то данная кэш-линия свободна и состояние остальных битов не важно. dirty означает, что кэш-линия хранит изменённые данные, которые ещё не записаны в память.

Параметры (вариант 1)

Кэш (продолжение)		
Ассоциативность	2 – **CACHE_WAY**	
Размер тэга адреса	10 бит – **CACHE_TAG_SIZE**	
Размер кэш-линии	16 байта – **CACHE_LINE_SIZE**	Размер полезных данных.
Кол-во кэш-линий	64 – **CACHE_LINE_COUNT**	
Память		
Размер памяти	512 Кбайт – **MEM_SIZE**	

Параметры (вариант 2)

Кэш (продолжение)		
Размер кэша	1 Кб – **CACHE_SIZE**	Размер полезных данных.
Размер кэш-линии	32 байта – **CACHE_LINE_SIZE**	Размер полезных данных.
Кол-во бит под хранение индекса набора	4 бита – **CACHE_SET_SIZE**	
Память		
Размер памяти	1 Мбайт – **MEM_SIZE**	

Параметры (вариант 3)

Кэш (продолжение)		
Размер кэша	2 Кб – **CACHE_SIZE**	Размер полезных данных.
Размер кэш-линии	16 байта – **CACHE_LINE_SIZE**	Размер полезных данных.
Кол-во бит под тэг адреса	8 бита – **CACHE_TAG_SIZE**	
Память		
Размер памяти	256 Кбайт – **MEM_SIZE**	(старое значение 128 Кбайт – **MEM_SIZE**)

Размерность шин

Шина	Обозначение	Размерность
A1, A2	**ADDR1_BUS_SIZE**, **ADDR2_BUS_SIZE**	Вычислить самостоятельно
D1, D2	**DATA1_BUS_SIZE, DATA2_BUS_SIZE**	16 бит
C1, C2	**CTR1_BUS_SIZE**, **CTR2_BUS_SIZE**	Вычислить

		самостоятельно

Время отклика

- 6 тактов – время, через которое в результате кэш попадания, кэш начинает отвечать.
- 4 такта – время, через которое в результате кэш промаха, кэш посылает запрос к памяти.
- MemCTR – 100 тактов

Время отклика – расстояние в тактах от первого такта команды до первого такта ответа.

Протокол обмена данными по шине

Команды и ответы на них передаются по шинам за несколько тактов подряд. Но между командой и ответом может быть произвольное кол-во тактов бездействия.

По шине A1 адрес передаётся за 2 такта: в первый такт tag+set, во второй - offset. По шине A2 передаётся адреса без части offset за 1 такт.

По шинам D (D1 и D2) в каждый такт передаётся по 16 бит данных, начиная с младших, little endian.

На линиях команд C (C1 и C2) значение держится всё время передачи команды или ответа.

В начальный момент времени (или после Reset) шиной 1 владеет CPU, а шиной 2 - Cache. После подачи команды и до окончания отправки ответа владение шиной переходит к Cache и MemCTR соответственно. Владение шиной означает какое устройство задает логические уровни на проводах шины.

Обозначения параметров в коде

В коде **команды** могут указываться как комментариями возле обозначения кода команды, так и являться значениями в перечисления (enum, подробнее):

- **C1_NOP**
- **C1_READ8**
- **C1_READ16**
- **C1_READ32**
- **C1_INVALIDATE_LINE**
- **C1_WRITE8**
- **C1_WRITE16**
- **C1_WRITE32**
- **C1_RESPONSE**

- **C2_NOP**
- **C2_READ_LINE**
- **C2_WRITE_LINE**
- **C2_RESPONSE**

Переменные, параметры, константы:

- **MEM_SIZE** – размер памяти
- **CACHE_SIZE** – размер кэша
- **CACHE_LINE_SIZE** – размер кэш-линии
- **CACHE_LINE_COUNT** – кол-во кэш-линий
- **CACHE_WAY** – ассоциативность
- **CACHE_SETS_COUNT** – кол-во наборов кэш-линий
- **CACHE_TAG_SIZE** – размер тэга адреса
- **CACHE_SET_SIZE** – размер индекса в наборе кэш-линий
- **CACHE_OFFSET_SIZE** – размер смещения
- **CACHE_ADDR_SIZE** – размер адреса

Детали реализации

Можно пользоваться любыми конструкциями и переменными языка описания Verilog и SystemVerilog.

Инициализация по умолчанию (также **RESET**):

- все кэш-линии в состоянии invalid
- в памяти данные инициализируются по следующему правилу:

```
module test #(parameter _SEED = 225526);
  integer SEED = _SEED;
  reg[7:0] a[0:99];
  integer i = 0;
  initial begin
    for (i = 0; i < 100; i += 1) begin
      a[i] = $random(SEED)>>16;
    end

    for (i = 0; i < 100; i += 1) begin
      $display("[%d] %d", i, a[i]);
    end

    $finish;
  end
endmodule
```

Алгоритм инициализации (включая SEED) запрещается менять.

Задача

Имеется следующее определение глобальных переменных и функций:

```
#define M 64
#define N 60
#define K 32
int8 a[M][K];
int16 b[K][N];
int32 c[M][N];

void mmul()
{
  int8 *pa = a;
  int32 *pc = c;
  for (int y = 0; y < M; y++)
  {
    for (int x = 0; x < N; x++)
    {
      int16 *pb = b;
      int32 s = 0;
      for (int k = 0; k < K; k++)
      {
        s += pa[k] * pb[x];
        pb += N;
      }
      pc[x] = s;
    }
    pa += K;
    pc += N;
  }
}
```

Сложение, инициализация переменных и переход на новую итерацию цикла, выход из функции занимают 1 такт. Умножение – 5 тактов. Обращение к памяти вида pc[x] считается за одну команду.

Массивы последовательно хранятся в памяти, и первый из них начинается с 0.

Все локальные переменные лежат в регистрах процессора.

По моделируемой шине происходит только обмен данными (не командами).

Определите процент попаданий (число попаданий к общему числу обращений) для кэша и общее время (в тактах), затраченное на выполнение этой функции.